Create Task Written Response Practice

AP CS Principle

This worksheet is designed to help students prepare for Part 2 of the AP CS P Exam. It contains AI generated open-ended questions that require students to provide written proof that they understand their code segments submitted for the Create Task. Each prompt is a slight adjustment of the original submission requirements for Component C of the Create Task found on pages 4 & 5 of the Student Handout booklet.

Prompt 1: Submit a program code segment that is a student-developed procedure that defines the procedure's name and return type (if necessary), contains and uses one or more parameters that have an effect on the functionality of the procedure, and implements an algorithm that includes sequencing, selection, and iteration.
(15 questions)

1. Procedure Definition:
Explain the significance of defining a procedure in a programming language. How does it contribute to code organization and reusability?

2. Return Type:
Elaborate on the concept of the return type in a procedure. Why might a procedure have a return type, and how does it affect the overall functionality of the program?

3. Parameters:
Describe the purpose and importance of parameters in a procedure. How do they enhance the flexibility and versatility of the code?

4. Parameter Effects:
Provide specific examples from your code segment where the parameters have a direct impact on the functionality of the procedure. How do different parameter values lead to different outcomes?

5. Algorithm Implementation:
Discuss the significance of implementing an algorithm in a programming task. How does it contribute to the overall structure and logic of the program?

6. Sequencing:
Identify and explain instances in your code where sequencing is crucial. How does the order of statements affect the program's execution?

7. Selection:
Point out parts of your code where conditional statements (selection) are utilized. Explain how these statements control the flow of the program based on certain conditions.

8. Iteration:
Discuss the presence and purpose of iteration (loops) in your code. How do loops contribute to the efficiency and repetitiveness of the procedure?

9. Combined Use of Sequencing, Selection, and Iteration:
Provide an example from your code where sequencing, selection, and iteration are combined to solve a specific problem. How do these elements work harmoniously to achieve the desired outcome?

10. Procedure Naming:
Justify the choice of the name for your procedure. How does it reflect the purpose and functionality of the code segment?

11. Parameter Types:
Distinguish between different types of parameters used in your procedure (e.g., input, output). How does each type contribute to the overall functionality?

12. Error Handling:
Discuss any error-handling mechanisms incorporated into your code. How do these mechanisms improve the robustness of your procedure?

13. Code Reusability:
Explain how your code segment promotes code reusability. Can the procedure be easily incorporated into other parts of the program or reused in different projects?

14. Algorithm Efficiency:
Evaluate the efficiency of the algorithm implemented in your code. Are there opportunities for optimization, and how might you enhance the overall performance?

15. Real-world Application:
Discuss a real-world scenario where the procedure you've written could be applied. How does the functionality of your code translate into practical use?

Prompt 2: Submit a second program code segment that shows where your student-developed procedure is being called in your program.
(10 Questions)

1. Procedure Call Placement:
Explain the specific location in your program where the student-developed procedure is called. Why did you choose this particular point in the code for the procedure call?

2. Parameter Values:
Provide examples of actual parameter values used when calling the procedure in your program. How do these values influence the behavior of the procedure during runtime?

3. Data Flow:
Trace the flow of data between the procedure call and the procedure definition. How are parameters passed, and how is data returned (if applicable)?

4. Conditional Procedure Calls:
Describe any conditions or criteria that determine whether the procedure is called in your program. How does the program logic decide when to invoke the procedure?

5. Multiple Procedure Calls:
If applicable, discuss instances where the procedure is called multiple times within your program. How does the repetition of the procedure call contribute to the overall functionality?

6. Return Values Handling:
If the procedure has a return type, explain how the returned value is utilized in the calling part of the program. Provide specific examples from your code.

7. Integration with Other Functions:
Discuss how the procedure call integrates with other functions or procedures in your program. How does this collaboration contribute to the program's overall functionality?

8. Error Handling in Procedure Calls:
Describe any error-handling mechanisms implemented when calling the procedure. How does the program respond if the procedure encounters unexpected issues during execution?

9. Procedure Call Optimization:
Evaluate the efficiency of the procedure call in your program. Are there opportunities for optimization, and how might you enhance the overall performance of the call?

10. Documentation of Procedure Calls:
Discuss the importance of documenting procedure calls in your code. How does proper documentation enhance the readability and maintainability of the program for yourself and other developers?

Prompt 3: Submit two program code segments you developed that contain a list (or other collection type) being used to manage complexity in your program. The first program code segment must show how data has been stored in the list. The second program code segment must show the data in the same list being used, such as creating new data from the existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.
(15 Questions)

Storing Data in the List:

1. List Initialization:
Describe how you initialized the list in the first code segment. Why did you choose a list, and what advantages does it offer in terms of data storage?

2. Data Types in the List:
Specify the types of data stored in the list. How did you ensure compatibility and coherence within the list?

3. Dynamic List Size:
Discuss whether the list size is fixed or dynamic. How does this choice impact the program's flexibility and resource utilization?

4. Data Modification in the List:
Explain any mechanisms in place for modifying data within the list. How does your program accommodate changes to the stored data?

5. List Manipulation Functions:
If applicable, list and explain any built-in or custom functions used for manipulating the contents of the list. How do these functions contribute to the program's functionality?

Using Data from the List:

6. Data Retrieval from the List:
Explain how data is retrieved from the list in the second code segment. What methods or techniques did you use to access specific elements?

7. Data Transformation:
Provide examples of how existing data in the list is transformed or used to create new data. How does this process contribute to achieving the program's goals?

8. Conditional Data Processing:
Discuss any conditions or criteria for processing data from the list. How do these conditions influence the flow of your program?

9. Iterating Through the List:
If applicable, describe how you iterate through the list in your code. How does this iteration contribute to achieving the program's objectives?

10. List-Based Decision Making:
Explain any decision-making processes in your program that rely on the contents of the list. How does the list play a role in determining the program's behavior?

Overall Program Purpose and Design:

11. List as a Complexity Management Tool:
Justify the choice of using a list as a complexity management tool in your program. How does it contribute to a more organized and maintainable code structure?

12. Alternative Data Structures:
Discuss whether you considered alternative data structures instead of a list. What factors influenced your decision, and how would the program be different with a different data structure?

13. Data Integrity and Validation:
Explain any measures taken to ensure data integrity within the list. How does your program handle invalid or unexpected data?

14. Scalability Considerations:
Consider the scalability of your program. How would the use of a list affect the program's performance as the size of the dataset increases?

15. Documentation of List Usage:
Discuss the importance of documenting how lists are used in your code. How does proper documentation enhance the understanding of your program for yourself and others?

These prompts aim to explore various aspects of using lists in programming, from initialization and manipulation to the practical application of the stored data in fulfilling the program's purpose.